REMARKS

The Advisory Action stated that the amendments made in the response to the last Final Office Action (a) raise new issues that would require further consideration and/or search and (b) would complicate effects of a potential Appeal.

By this amendment, Claims 10, 14, 21, 25, 32, 36, 43, 46, and 53 are amended. Claims 1-9, 11, 13, 15, 22, 24, 26, 33, 35, 37, 45, and 47 are canceled, and no claims are added. Hence, Claims 10, 12, 14, 16-21, 23, 25, 27-32, 34, 36, 38-44, 46, and 48-56 are pending in the application.

The amendments to the claims as indicated herein do not add any new matter to this application. Instead, the amendments to the independent claims incorporate the subject matter of the corresponding canceled claims. For example, Claim 10 now includes the subject matter of canceled Claims 11, 13, and 15. Canceled Claim 15 depended on canceled Claim 13, which depended on canceled Claim 11, which depended on Claim 10. Claim 10 contains the same subject matter and scope of canceled Claim 15.

Each issue raised in the Final Office Action mailed April 7, 2008 is addressed hereinafter.

I. SOLE REJECTION

Claims 1-56 stand rejected under 35 U.S.C. § 103(a) as allegedly being unpatentable over U.S. Patent No. 5,008,814 issued to Mather ("*Mathur*"), in view of U.S. Patent Application No. 2005/0198629 to Vishwanath ("*Vishwanath*"). This rejection is respectfully traversed.

A. CLAIM 10

Claim 10 now recites:

> A method of software loading and initialization in a distributed network of nodes, the method comprising:
> persistently storing, in a first storage of a master node, a plurality of software packages and a plurality of boot images, wherein the plurality of software

packages and the plurality of boot images will be used by the nodes in the distributed network;

persistently storing, in a second storage of the master node, software version information and node type information for each node in the distributed network;

receiving, at the master node, a request for a boot image and software packages from a node, in the distributed network, that is performing an initial boot;

based on the request, the master node determining a boot image of the plurality of boot images and one or more software packages of the plurality of software packages to extract from the first storage;

the master node extracting the boot image and the one or more software packages from the first storage;

delivering, to the node, the boot image and the one or more software packages, wherein said node:

(a) stores the boot image and the one or more software packages in its local persistent storage,

(b) **extracts software version information from the one or more software packages** and stores the software version information in the local persistent storage,

(c) reboots and executes the boot image stored in the local persistent storage, and

(d) **verifies the software version information with said master node**;

if said node does not have the correct software versions, then the master node retrieving correct software packages from the first storage and sending the correct software packages to said node, wherein said node stores the correct software packages in the local persistent storage and completes booting by executing the correct software packages stored in the local persistent storage. (emphasis added)

At the least the above-bolded features of Claim 10 are not taught or suggested by *Mathur* or *Vishwanath*, either individually or in combination.

1. The cited art fails to teach or suggest what happens if the software versions of software packages that a node receives are not correct

On page 7, the Final Office Action cites the consistency check of FIG. 2 of *Mathur* for allegedly disclosing:

> if said node does not have the correct software versions, then the master node retrieving correct software packages from the first storage and sending the correct software packages to said node, wherein said node stores the correct software packages in the local persistent storage and completes booting by executing the correct software packages stored in the local persistent storage

as recited in Claim 10. This is incorrect. In fact, this cited portion of *Mathur* **teaches the opposite** of this feature of Claim 10. The check consistency step is referenced in steps 202 and 207 of FIG. 2. As FIG. 2 depicts and as the corresponding text teaches, in both cases where the check consistency fails, old software is re-used. Specifically, contents of an "old" non-volatile storage device on a destination node (i.e., the alleged "node" of Claim 10) are copied either to a "dirty" non-volatile storage device (referred to as a "loadback process") of the destination node or to a "trial" non-volatile storage device (referred to as a "cutback process") of the destination node. What's more, the check consistency of step 202 occurs at the source node (i.e., the alleged "master node" of Claim 10) and occurs before any communication has even been made with any destination node. Fundamentally, the destination node of *Mathur* does not **receive another set of software packages from the source node** if the destination node does not have the correct software versions.

2. *The cited art fails to teach or suggest that a node that receives software packages verifies software version information with the master node*

On page 6, the Final Office Action also cites steps 204 and 211 of FIG. 2 of *Mathur* for allegedly disclosing "wherein said node…verifies the software version information with said master node," as recited in Claim 10. This is also incorrect. In step 204, the source node of *Mathur* transmits new software to the destination node. In step 211, the destination node performs an update process which comprises copying the content of a "trial" non-volatile storage device to all of the other non-volatile storage devices of the destination node. In neither step does the **destination node verify software version information with the source node**, as Claim 10 requires. Indeed, the entire *Mathur* reference lacks any teaching or suggestion of a destination node verifying software version information with a source node, much less

performing this verification step **after** the destination node receives software packages from the source node.

The Advisory Action asserts:

> As for the 'extracting version information' limitation, the verifying step as proffered based on Mathur involves a final step of receiving of a correct version AFTER a consistency check is made at the destination node, which includes checksum verification (which includes extracting for this CRC to be computed in terms of version validating information in regard to a supposedly new software) via relay action with an operator (see col. 5-6); clearly, the destination node has way to communicate with the master about correctness of this new version (version information) intended for distribution, leading to completing its reception at step 204. (emphasis added)

This is incorrect for at least two reasons. First, a destination node of *Mathur* receives software before the destination node performs a consistency check. It is not possible otherwise. Second, although the destination node does perform a checksum compare between the packets it receives and the checksum received from the source node, the destination node does not perform a version validation, as alleged.

The Final Office Action appears to interpret "software version information" as any information about the software that the master node delivers to the receiving node. Such an interpretation is unreasonably broad. MPEP § 2111.01(I) states that "the words of the claim **must** be given their plain meaning unless the plain meaning is inconsistent with the specification" (emphasis added). Also, MPEP § 2111.01 (III) states that the "ordinary and customary meaning of a claim term is the meaning that the term would have to a person of ordinary skill in the art in question at the time of the invention." It is clear that "software version information" means something more specific than simply "software information." If Applicants intended to cover any information about software, then Applicants would have used "software

information" instead. Moreover, according to the ordinary and customary meaning of "software version information" of a software package, such information at least indicates the version of the software package relative to other versions of the software package.

MPEP § 2111.01 (III) further states that the "ordinary and customary meaning of a term may be evidenced by a variety of sources, including 'the words of the claims themselves…[and] the remainder of the specification.'" The specification makes clear that a software package contains "version information, dependency information, and other metadata information pertaining to the software in the package." (see, e.g., paragraph 78). Therefore, "software version information" is more specific than just information about software.

3. The cited art fails to teach or suggest that software version information is extracted from software packages that are delivered to a node from the master node

On page 9, the Final Office Action cites col. 7, lines 32-53; col. 6, lines 41-54; and col. 3, lines 29-53 of *Mathur* for allegedly disclosing "wherein said node…extracts software version information from the one or more software packages and stores the software version information in the local persistent storage," as recited in Claim 10. This is incorrect.

Col. 7, lines 32-53 of *Mathur* teaches a target or destination node performing a cutover process, a consistency check, and an IPL (i.e., an initial program boot). If the IPL fails, then the destination node performs a cutback process to revert to older software it was executing.

Col. 6, lines 41-54 of *Mathur* teaches a destination node receiving data packets pertaining to new software that is distributed from a source node. The destination node computes a checksum of from the packets and compares that checksum to a checksum transmitted from the source node. If the checksums match, then the destination node sends an acknowledgement to the source node. If the source node does not receive an acknowledgement, then the source node will retransmit one or more packets to the destination node

Col. 3, lines 29-53 of *Mathur* teaches that "[network] topological information, which is used for communicating and routing messages between nodes, resides in a CPU's working memory." System software is stored as a number of modules in a non-volatile storage device of a node. This cited portion finally teaches that the "network topological information is periodically maintained and updated to reflect changes in the configuration of the network."

Nothing in these cited portions teaches or suggests that a destination node extracts software version information from software packages that the destination node receives from a source node.

In rejecting this feature of present Claim 10, the Final Office Action further states, "Note: packet reception of new software and for ILP for storage at node reads on extraction of package received based on version, checksum, packet time and topology of nodes etc." There are at least two problems with this assertion. First, Claim 10 does not recite extracting a package based on version. Instead, Claim 10 recites that a node extracts software version information from one or more software packages. Second, it is **unreasonable** to interpret the mere packet reception of new software at a destination node and the destination node performing an IPL as including the **specific limitation** that a node extracts software **version** information from one or more software packages.

Fundamentally, *Mathur* lacks any teaching or suggestion that a destination node **extracts** software version information (much less from software packages that are sent to the destination node) and stored in persistent storage of the destination node. No destination node in *Mathur* is concerned with software version information of new software that it receives. Therefore, it is not surprising that *Mathur* fails to teach or suggest this feature of present Claim 10.

Vishwanath is not cited for teaching or suggesting any of the features of Claim 10 discussed above.

Based on the foregoing, *Mathur* and *Vishwanath* fail to teach or suggest, both individually and in combination, all the features of Claim 10. Therefore, Claim 10 is patentable over the cited art. Reconsideration and withdrawal of the rejection of Claim 10 under 35 U.S.C. § 103(a) is therefore respectfully requested.

B. CLAIMS 21, 32, AND 43

Each of independent Claims 21, 32, and 43 is either a computer-readable storage medium, an apparatus, or a system claim. Each of Claims 21, 32, and 43 recites the features discussed above that distinguish present Claim 10 over *Mathur* and *Vishwanath*. Therefore, each of Claims 21, 32, and 43 is patentable over the cited art for at least the reasons given above for Claim 10.

C. CLAIMS 16, 27, 38, AND 48

Each of Claims 16, 27, 38, 48 recites that the master node has the ability to categorize nodes into classes where all of the nodes in a particular class of nodes have the same software configuration. The Final Office Action cites col. 5, lines 3-27 and col. 3, lines 29-53 of *Mathur* for allegedly disclosing this feature. The Final Office Action further states, "Note: structure representing grouping of nodes according to some particular privilege configuration reads on class of nodes of same configuration but different processor." This is incorrect.

The applicable cited portion of *Mathur* states:

> Advantageously, the system is implemented so that a distribution command is recognizable by a controller 100 only when it is issued with predefined **access privileges**. Furthermore, **such privileges** may be defined with an hierarchical structure so that the privilege required for a distribution process depends upon the importance of the software.
>
> (col. 5, lines 21-27; emphasis added)

Thus, the privileges referred to in *Mathur* and in the Final Office Action are **access** privileges that are used to determine whether a distribution command (which identifies a source node and one or more destination nodes) will be recognizable, and thus executable. Contrary to the assertion in the Final Office Action, *Mathur* does **not** suggest a "structure representing grouping **of nodes**" (emphasis added). The Final Office Action fails to cite any portion of *Mathur* for teaching or suggesting such a structure or grouping. In fact, the cited portion of *Mathur* teaches the prior art approach of requiring a human user to identify each destination node of a software update (see col. 5, lines 1-3, 6-7, and 12-14). *Vishwanath* is not cited for teaching or suggesting this feature of Claims 16, 27, 38, and 48.

The Advisory Action alleges:

> The scheme for distribution by the controller in Mathur operates via access privileges basis, such basis being a hierarchized structure, to enable only a certain amount of nodes in said structure elligible for receiving new software; i.e. privilege status whereby these nodes (Mathur: col. 5, li. 12-17) receive the software reads on a class of nodes having same configuration. (emphasis added)

This is incorrect. *Mathur* does not relate this hierarchical structure (that defines access privileges) with the destination nodes. The hierarchical structure has nothing to do with the destination nodes. Instead, the hierarchical structure is merely used to determine whether software will be distributed at all. Therefore, it is incorrect to assert that "Mathur operates via access privileges basis…to enable only a certain amount of nodes in said structure elligible for receiving new software" (emphasis added). Even if this hierarchical structure referred to destination nodes, *Mathur* still fails to mention any about the configuration of nodes.

Based on the foregoing, *Mathur* and *Vishwanath* fail to teach or suggest, both individually and in combination, all the features of Claims 16, 27, 38, and 48. Therefore, each of Claims 16, 27, 38, and 48 is patentable over the cited art. Reconsideration and withdrawal of the

rejection of Claims 16, 27, 38, and 48 under 35 U.S.C. § 103(a) is therefore respectfully requested.

D. REMAINING DEPENDENT CLAIMS

The dependent claims not discussed thus far are dependent claims, each of which depends (directly or indirectly) on one of the independent claims discussed above. Each of the dependent claims is therefore patentable over the cited art for at least the reasons given above for the claim on which it depends. In addition, each of the dependent claims introduces one or more additional limitations that may independently render it patentable. However, due to the fundamental differences already identified and to expedite the positive resolution of this case, a separate discussion of those limitations is not included at this time. The Applicants reserve the right to further point out the differences between the cited art and the novel features recited in the dependent claims.

II. CONCLUSIONS & MISCELLANEOUS

For the reasons set forth above, all of the pending claims are now in condition for allowance. The Examiner is respectfully requested to contact the undersigned by telephone relating to any issue that would advance examination of the present application.

A petition for extension of time, to the extent necessary to make this reply timely filed, is hereby made. If applicable, a law firm check for the petition for extension of time fee is enclosed herewith. If any applicable fee is missing or insufficient, throughout the pendency of this application, the Commissioner is hereby authorized to any applicable fees and to credit any overpayments to our Deposit Account No. 50-1302.

Respectfully submitted,
HICKMAN PALERMO TRUONG & BECKER LLP

Dated: August 6, 2008

/DanielDLedesma#57181/
Daniel D. Ledesma
Reg. No. 57,181

2055 Gateway Place Suite 550
San Jose, California 95110-1083
Telephone No.: (408) 414-1080 ext. 229
Facsimile No.: (408) 414-1076